BENCHMARKS WITH POINTS ON ELLIPTIC CURVES

Josef Brychta
Doctoral Degree Programme (1), FEEC BUT
E-mail: xbrych07@vutbr.cz

Supervised by: Radek Fujdiak
E-mail: fujdiak@vutbr.cz

Abstract: This paper deals with benchmarks of generating random points and arithmetic operations with points on elliptic curves. These benchmarks were implemented on the hardware platforms Raspberry Pi Zero/1/2/3. The primary goal was to measure usage of the system resources and energetic consumption of that hardware platform. Specifically, it was memory and time consumption of Raspberry. In total, 48 elliptic curves of standards NIST, BRAINPOOL, GOST and FRP were measured.

Keywords: OpenSSL, LibECC, Python, generating/multiplying random points, Double-and-Add, Montgomery ladder, wNAF, elliptic curves, NIST, BRAINPOOL, GOST, FRP, Raspberry Pi.

1 INTRODUCTION

Cryptography, especially nowadays, is used in all areas which are related to computers or networks. The main usages of cryptography are in particular internet banking, credit cards, file encryption, discs or just secure access to the websites. One way how to move cryptography to the next level is to use elliptic curves, that allow the use of random or pseudo-random variables in cryptographic protocols. For this purpose, the use of random points on elliptic curves and the use of arithmetic operations with points (such as addition, multiplication and so on.) on elliptic curves [1, 2]. Modern cryptographic libraries allow working with elliptic curves as a side function, or as the complex cryptographic protocols which work on elliptic curves. For example some of the well-known algorithms: ECDH – Elliptic curve Diffie Hellman and ECDSA – Elliptic Curve Digital Signature Algorithm [3, 4, 5]. The motivation for writing this article was the creation of benchmarks for measuring randomly generated points and arithmetic operations with these points on elliptic curves to determine their real efficiency on the platforms. The hardware platforms used here were the Raspberry Pi Zero/1/2/3 devices, which datasheets can be found on the production pages [6]. Firstly, the article introduces readers with the used cryptographic libraries, their brief description and functionality. Secondly, acquaints readers with the principle of benchmark measurement, the main measured parameters, and ultimately presents the benchmark results, their discussion and conclusion of this paper.

2 LIBRARIES AND ELLIPTIC CURVES

In this section, selected libraries and elliptic curves, where measurements of random point generating and arithmetic operations were performed are described.

OpenSSL works with cryptography on elliptic curves too, beyond its widespread use in the field of security. This library provides an extensive set of functions for performing operations on elliptic curves with finite fields. Arithmetic operations on elliptic curves allow too. OpenSSL is the complex library for the implementation of ECDSA and ECDH protocols. Therefore, the primary focus is the implementation of these algorithms [7].

LibECC on the other hand, opposes OpenSSL exclusively by cryptography on elliptic curves. This library provides an extensive set of functions to perform elliptic curve's operations in the finite field.

The library provides the appropriate tools to perform test benchmarks. Other uses of this library include ECDH protocol implementation capabilities [8].

Python is an interpretive high-level programming language for general programming. One of the possibilities of this programming language is the possibility of precisely defining the mathematical function (such as elliptic curves). Further, the declaration of the point on the elliptic curve and the related actions. It was included here for comparing performance with specialized libraries.

Elliptic curves are selected curves (NIST and BRAINPOOL standards, marginally also GOST and FRP) are from interval from 112 to 512 bits. The parametric designation of elliptic curves in the results is R, K, T. Parametric designation R means an elliptic curve with randomly selected parameters in the range of bits equal to the field size above which the elliptic curve is located. Parametric designation K means Koblitz elliptic curve, which is controlled by its parametrization. Finally, the T parameter refers to Twisted Edwards elliptic curves, which are also controlled by their parametrization. More about this parametrization can be read in the official documentation of the NIST standards (SECP, SECT) [9] and BRAINPOOL [10].

3 BENCHMARKS AND MEASUREMENTS

The measurement was performed using a Python measurement algorithm which is using the PSutil library. It was also necessary to program the computational operations, meaning it is to generate points on elliptic curves and arithmetic operations with these points. Generating random points on the elliptic curve was programmed using internal algorithms of the libraries which allow to customize point generation by the required criteria. The algorithm worked by generating points in a given group in which the elliptic curve was located and verifying whether it was on that curve. Therefore, generating takes as long as a point that corresponds to it is not generated. A specific measurement algorithm measures the average of thousands of generated points on the elliptic curve. Furthermore, arithmetic operations with already generated points on the elliptic curve were programmed. This means as addition, multiplication and so on. Multiplication has been measured in the article because it is used most often in cryptographic protocols and is the most demanding. It was realized by taking one coordinate, such as x, a random point on an elliptic curve of magnitude n. Thus, the n-bit number (n is the group size) and our constant for the given curve, i.e. A (A = xo size n). This constant was the same for all measurements of the given curve across platforms. Constant A was generated for each curve size differently. The arithmetics multiplication of points on the elliptic curve was carried out using the basic scalar method Double-and-Add, the scalar method wNAF and the scalar method Montgomery ladder. The differences between these methods are purely mathematical and they are currently being used together with other methods [11]. The current measurement algorithm works as follows. Each test benchmark (computational operations) is assigned as a PPID identifier (further to a unique process ID, each process is assigned a parent process ID (PPID) that tells which process started it. The PPID is the PID of the process's parent) immediately after the start of the operation. This algorithm runs in a certain cycle. Specifically, 1000 repetitions. After the cryptographic libraries are loaded and the computational operations of these libraries are started, individual measurements are started. Once the computational operations run, the measurements are also terminated. The results of the individual measurements are recorded in the file. The values are measured in milliseconds and averaged. Flowchart of measurement algorithm in Figure 1.

4 MEASURED VALUES

Here we can see the measured values of random point generating and multiplication of randomly generated points on elliptic curves of SECP, BRAINPOOL, GOST and FRP in the LibECC library. The values are written in Table 1. The measurement shows some dependence in proportion to the

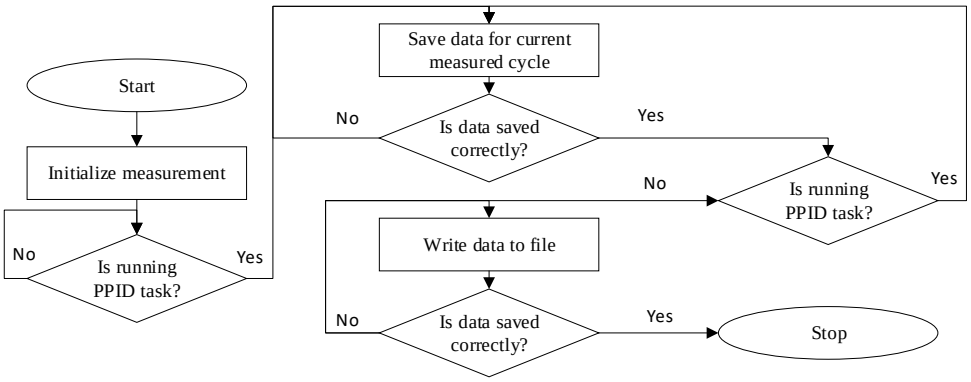

Figure 1: Flowchart of measurement algorithm.

Table 1: Measured values in LibECC library.

-	Time (ms)											
-	Raspberry Pi 3			Raspberry Pi 2			Raspberry Pi Zero			Raspberry Pi 1		
Curve	1	2	3	1	2	3	1	2	3	1	2	3
SECP												
192R1	2.185	13.104	26.557	3.636	21.601	44.908	6.933	40.917	80.884	4.927	31.084	59.276
224R1	3.557	20.606	44.395	6.025	34.153	74.212	10.604	63.977	134.592	8.023	48.011	99.910
256R1	4.179	23.648	46.037	7.030	39.316	76.872	12.620	73.307	136.913	10.069	55.527	102.985
384R1	12.083	61.090	111.587	19.958	101.986	183.921	37.309	187.708	324.403	26.306	141.964	242.413
521R1	30.995	158.933	289.114	53.005	267.278	475.409	96.303	475.414	853.945	70.905	359.115	629.021
BRAINPOOL												
P224R1	3.675	20.652	44.343	6.097	34.350	73.834	11.734	60.841	134.703	9.248	48.179	97.159
P256R1	4.211	23.601	45.962	7.176	39.261	76.646	12.997	73.112	135.978	9.626	55.528	99.817
P384R1	11.414	60.974	111.297	19.474	102.157	184.984	36.639	187.647	333.615	26.471	140.892	247.983
P512R1	24.574	128.204	222.814	42.312	215.234	368.641	76.518	390.125	662.321	57.872	292.574	483.166
GOST												
256	17.695	23.526	45.870	29.903	39.187	76.246	53.358	69.152	137.810	40.521	54.737	102.343
512	25.742	127.869	237.269	43.622	214.677	392.419	76.583	388.960	683.528	56.027	293.821	522.168
FRP												
256V1	4.153	23.592	45.969	7.105	39.208	77.364	12.881	73.225	138.597	11.380	55.523	103.410

increasing time required to generate a point and multiplication a point on the size of a given curve. From the least time-consuming SECP curves, through the FRP, BRAINPOOL and the most sophisticated SECT curves. There are the values of multiplication of randomly generated points on elliptic curves of SECP and BRAINPOOL in the Python library. The values are written in Table 2.

Table 2: Measured values in Python library.

-	Time (ms)			
Curve	Raspberry Pi 3	Raspberry Pi 2	Raspberry Pi Zero	Raspberry Pi 1
SECP				
192K1	69.059	113.531	298.502	349.552
192R1	70.463	115.833	290.253	351.543
224K1	93.349	154.472	365.240	480.848
224R1	95.120	157.446	356.378	461.448
256K1	125.032	208.003	464.327	613.763
256R1	127.318	217.121	463.424	610.232
384R1	290.350	488.917	1210.727	1360.019
521R1	558.108	958.343	2184.007	2712.240
BRAINPOOL				
P160R1	47.907	82.888	221.820	294.222
P192R1	71.021	122.191	305.478	395.037
P224R1	93.448	157.934	359.223	541.504
P256R1	125.364	205.897	467.335	720.090
P320R1	194.608	335.035	733.409	1049.705
P384R1	303.914	491.724	1022.278	1452.071
P512R1	549.072	903.787	2131.401	2794.490

The measurement shows some dependence in proportion to the increasing time required to generate a point on the size of a given curve. From the least time-consuming SECP curves, through the FRP,

BRAINPOOL and the most sophisticated SECT curves. The measured values of random point generating and multiplication on elliptic curves of SECP, BRAINPOOL, GOST and FRP in the OpenSSL library are written in Table 3.

Table 3: Measured values in OpenSSL library.

-	Time (ms)							
-	Raspberry Pi 3		Raspberry Pi 2		Raspberry Pi Zero		Raspberry Pi 1	
Curve	1	2	1	2	1	2	1	2
SECP								
112R1	1.010	0.912	1.559	1.392	3.405	2.857	4.629	4.130
112R2	1.024	1.867	1.576	2.859	3.587	6.065	4.715	8.660
128R1	1.112	2.896	1.715	4.436	3.578	9.309	5.109	13.341
128R2	1.133	3.964	1.777	6.099	3.690	12.758	5.632	18.262
160K1	1.769	5.611	2.797	8.672	5.987	18.236	8.934	26.427
160R1	1.614	7.103	2.514	10.968	5.227	23.023	7.437	33.161
160R2	1.616	8.597	2.518	13.290	5.078	27.671	7.350	39.873
192K1	2.389	10.855	3.951	17.045	8.169	35.298	11.816	51.137
224K1	3.180	13.877	5.321	22.117	10.946	45.595	15.593	66.120
224R1	2.850	16.574	4.720	26.580	9.742	54.339	13.751	78.552
256K1	4.199	20.600	5.999	32.349	13.666	67.363	19.746	97.732
384R1	8.874	29.175	14.865	46.836	27.434	94.474	39.592	136.282
521R1	19.349	48.083	33.420	79.955	60.699	154.408	86.438	219.871
SECT								
113R1	1.402	1.365	1.600	1.392	3.428	3.291	5.082	4.871
113R2	1.406	2.722	1.607	2.859	3.417	6.529	5.398	9.642
131R1	2.561	5.297	2.795	4.436	6.180	12.599	9.365	18.803
131R2	2.638	7.871	2.857	6.099	6.344	18.829	9.430	27.825
163K1	3.354	11.168	3.680	8.672	8.064	26.553	11.755	39.404
163R1	3.607	14.732	3.894	10.968	8.615	35.079	12.631	51.568
163R2	3.626	18.301	3.916	13.290	8.791	43.451	13.150	64.069
193R1	4.683	22.855	4.729	17.045	11.190	54.037	16.134	79.647
193R2	4.537	27.403	4.626	22.117	10.629	64.516	15.598	95.221
233K1	6.286	33.679	6.277	26.580	14.593	79.013	21.863	116.606
233R1	6.948	40.609	6.813	32.349	16.470	94.971	23.877	140.103
239K1	6.443	47.058	6.428	46.836	15.118	109.801	22.092	162.036
283K1	11.575	58.714	11.371	79.955	27.020	136.721	39.883	198.530
283R1	12.955	71.726	12.420	71.103	30.496	166.221	44.364	239.284
409K1	26.247	98.187	23.248	94.002	61.165	226.094	82.758	320.861
409R1	30.014	128.309	26.219	119.805	69.676	293.841	93.140	420.041
571K1	61.400	189.930	54.230	173.672	133.040	431.807	189.659	618.178
571R1	70.901	260.024	61.326	234.549	151.954	585.000	230.124	850.972
BRAINPOOL								
P160T1	1.620	3.052	2.448	2.475	5.026	9.574	7.196	13.907
P160R1	1.698	1.574	2.626	4.790	5.426	4.875	7.752	7.185
P192T1	2.179	7.195	3.470	11.692	7.053	22.638	10.051	33.130
P192R1	2.310	5.189	3.747	8.380	7.396	16.374	10.427	23.757
P224T1	2.870	12.743	4.616	20.931	9.154	40.513	13.031	59.014
P224R1	3.081	10.073	5.034	16.539	10.176	31.970	14.430	46.720
P256T1	3.779	20.156	5.346	31.468	11.602	63.582	17.328	92.537
P256R1	4.126	16.614	5.747	26.426	12.883	52.557	18.274	76.548
P320T1	6.007	32.129	10.034	52.100	18.695	101.096	27.125	146.643
P320R1	6.633	26.455	11.360	42.392	20.952	83.279	29.248	121.098
P384T1	8.919	50.189	14.593	82.573	27.097	156.017	39.348	226.702
P384R1	9.996	41.694	16.512	68.291	31.223	130.928	43.727	188.768
P512T1	17.499	86.289	25.864	137.637	54.354	258.467	77.879	387.540
P512R1	19.963	69.440	29.478	111.767	62.178	210.662	87.791	312.774

In this case, the measurement shows some dependence in proportion to the increasing time required to generate a point on the size of a given curve. From the least time-consuming SECP curves, through the FRP, BRAINPOOL and the most sophisticated SECT curves.

5 CONCLUSION

This paper dealt with benchmarks for generating random points and arithmetic operations with points on elliptic curves. These benchmarks have been implemented on the Raspberry Pi Zero/1/2/3 hardware platforms. Concretely, the time demands of these operations. A total of 48 elliptic curves of NIST and BRAINPOOL standards were measured, marginally also GOST and FRP. Finally, all libraries were compared to the most demanding operation by multiplying randomly generated points on elliptical curves on Raspberry Pi 3 device that was the fastest of the devices being measured.

This chapter compares the measured values of multiplying randomly generated points on elliptical curves of SECP and BRAINPOOL standards because of article capacity and also because these two standards were implemented in all libraries on all measured devices. The remaining GHOST and FRP

standards were implemented only in LibECC library. It can be seen in tables that multiplication by the Double-and-Add scalar method in Python library is considerably slower than other measurements. Only slightly slower performance was measured with LibECC library with Double-and-Add scalar multiplication methods and Montgomery ladder versus OpenSSL library. Further, LibECC library shows faster multiplication by using Montgomery ladder scalar multiplication. The fastest is wNAF scalar multiplication method, which is only implemented in OpenSSL library. Based on more extensive measurements beyond the published values in this article, the addition of randomly generated points on the elliptical curve was faster in LibECC library than this in OpenSSL library.

REFERENCES

[1] MAO, W. *Modern Cryptography: Theory and Practice.*. 2004, ISBN 1-306-6943-1. [cit. 2. 13. 2019].

[2] HANKERSON, D; MENEZES, A; VANSTONE, S. *Guide to Elliptic Curve Cryptography*. 2004, ISBN 978-0387952734 [cit. 2. 13. 2019].

[3] ANOOP, M. *ECC* [online]. 2015, last revised 2015 [cit. 2. 13. 2019]. Available from URL: <https://www.johannes-bauer.com/compsci/ecc>.

[4] JOHNSTON, O. *A Discrete Logaritm Attack On Elliptic Curves* [online]. 2010, last revised 2010 [cit. 2. 14. 2019]. Available from URL: <https://eprint.iacr.org/2010/575.pdf>.

[5] BOS, W; HALDERMAN, A; HENINGER, N. *Elliptic Curve Cryptography in Practice* [online]. 2013, last revised 2013 [cit. 2. 14. 2019]. Available from URL: <https://eprint.iacr.org/2013/734.pdf>.

[6] *Raspberry Pi 3 Model B 64-bit 1GB RAM* [online]. 2016, last revised 2016 [cit. 2. 15. 2019]. Available from URL: <http://rpishop.cz/kategorie/283-raspberry-pi-3-model-b-64-bit.html>.

[7] *OpenSSL library* [online]. 2017, last revised 2017 [cit. 2. 16. 2019]. Available from URL: <https://github.com/openssl/openssl>.

[8] *libecc library* [online]. 2017, last revised 2017 [cit. 2. 16. 2019]. Available from URL: <https://github.com/ANSSI-FR/libecc>.

[9] *SEC 2: Recommended Elliptic Curve Domain Parameters* [online]. 2010, last revised 2010 [cit. 3. 10. 2019]. Available from URL: <http://www.secg.org/sec2-v2.pdf>.

[10] *Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation* [online]. 2010, last revised 2017 [cit. 3. 10. 2019]. Available from URL: <https://tools.ietf.org/html/rfc5639>.

[11] MENEZES, A; OORSCHOT P, VANSTONE S. *Handbook of Applied Cryptography*. 1997, CRC Press [cit. 3. 10. 2019]. Available from URL: <https://zygoteiot.wordpress.com/2017/03/16/a-primer-to-elliptic-curve-diffie-hellman>.

[12] VUT, BRNO. *Proceedings of the 24th Conference STUDENT EEICT 2018* [online]. 2018, last revised 2018, ISBN 978-80-214-5614-3. [cit. 3. 10. 2019].

[13] FUJDIAK, R. *ANALYSIS AND OPTIMIZATION OF DATA COMMUNICATION FOR TELEMETRIC SYSTEMS IN ENERGY)*. 2017 [cit. 3. 10. 2019]. Available from URL: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=157492>.